ITU - Telecommunication Standardization Sector **Temporary Document ??-E(WP2/16)**

STUDY GROUP **16**

Santiago, 18-28 May 1999

Question: 12

SOURCE*: RAPPORTEUR Q12/16 (SAKAE OKUBO)

TITLE: PROPOSED DRAFT AMENDMENT 7 TO ITU-T RECOMMENDATION H.222.0 | ISO/IEC 13818-1

This document proposes a draft for Amendment 7 to the common text ITU-T Recommendation H.222.0 | ISO/IEC 13818-2 "Information technology - generic coding of moving pictures and associated audio information: video."

This amendment specifies carriage of ISO/IEC 14496 (MPEG-4) content over H.222.0|ISO/IEC 13818-1 (MPEG-2) System streams. This will enable multiple ways to use MPEG-4 content within MPEG-2 based applications, such as :

- MPEG-4 Elementary Streams in MPEG-2. In this case one or more MPEG-4 elementary streams can be carried in an MPEG-2 multiplex, for example an MPEG-4 encoded complementary speech channel within an MPEG-2 coded program.
- MPEG-4 Scenes in MPEG-2. In this case MPEG-4 scenes, i.e. MPEG-4 Elementary Stream and System data, are carried, for example to enhance MPEG-2 broadcast services with MPEG-4 interactive applications. The MPEG-4 data is intended to be decoded by MPEG-2 receivers enhanced with MPEG-4 functionality. To allow for decoding at these receivers, a set of constraints specific for carriage over MPEG-2 are applied to the MPEG-4 data that is carried.

Items 1) through 5) list proposed addition to the syntax as marked with the double line on the right end and Items 6) through 8) list additional texts.

In ISO/IEC JTC 1 SC 29/WG 11, this is currently in the status of Final Proposed Draft Amendment (FPDAM) being balloted and will be elevated to Final Draft Amendment (FDAM) in October 1999 and subsequently to Amendment (AMD) in December 1999.

It is proposed that this amendment be determined at the SG 16 meeting in May 1999 to achieve synchronized approval between ITU-T and ISO/IEC JTC 1.

*** Contact:** Sakae Okubo
Telecommunications Advancement Organization of Japan
Tokyo, Japan

Tel: +81 3 5286 3830
Fax: +81 3 5287 7287
Email: okubo@giti.or.jp

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY -- GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION: SYSTEMS

AMENDMENT 7

1) Replace table 2-17 in subclause 2:

“

Table 2-17 -- PES packet

Syntax	No. of Bits	Mnemonic
PES_packet() {		
packet_start_code_prefix	**24**	**bslbf**
stream_id	**8**	**uimsbf**
PES_packet_length	**16**	**uimsbf**
if(stream_id != program_stream_map		
&& stream_id != padding_stream		
&& stream_id != private_stream_2		
&& stream_id != ECM		
&& stream_id != EMM		
&& stream_id != program_stream_directory		
&& stream_id != DSMCC_stream		
&& stream_id != ITU-T Rec. H.222.1 type E_stream		
&& stream_id != ISO/IEC14496-1_FlexMux_stream) {		
'10'	**2**	**bslbf**
PES_scrambling_control	**2**	**bslbf**
PES_priority	**1**	**bslbf**
data_alignment_indicator	**1**	**bslbf**
copyright	**1**	**bslbf**
original_or_copy	**1**	**bslbf**
PTS_DTS_flags	**2**	**bslbf**
ESCR_flag	**1**	**bslbf**
ES_rate_flag	**1**	**bslbf**
DSM_trick_mode_flag	**1**	**bslbf**
additional_copy_info_flag	**1**	**bslbf**
PES_CRC_flag	**1**	**bslbf**
PES_extension_flag	**1**	**bslbf**
PES_header_data_length	**8**	**uimsbf**
if (PTS_DTS_flags =='10') {		
'0010'	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
if (PTS_DTS_flags ==‘11’) {		
'0011'	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**

'0001'	4	bslbf
DTS [32..30]	3	bslbf
marker_bit	1	bslbf
DTS [29..15]	15	bslbf
marker_bit	1	bslbf
DTS [14..0]	15	bslbf
marker_bit	1	bslbf
}		
if (ESCR_flag=='1') {		
reserved	2	bslbf
ESCR_base[32..30]	3	bslbf
marker_bit	1	bslbf
ESCR_base[29..15]	15	bslbf
marker_bit	1	bslbf
ESCR_base[14..0]	15	bslbf
marker_bit	1	bslbf
ESCR_extension	9	uimsbf
marker_bit	1	bslbf
}		
if (ES_rate_flag == '1') {		
marker_bit	1	bslbf
ES_rate	22	uimsbf
marker_bit	1	bslbf
}		
if (DSM_trick_mode_flag == '1') {		
trick_mode_control	3	uimsbf
if (trick_mode_control == fast_forward) {		
field_id	2	bslbf
intra_slice_refresh	1	bslbf
frequency_truncation	2	bslbf
}		
else if (trick_mode_control == slow_motion) {		
rep_cntrl	5	uimsbf
}		
else if (trick_mode_control == freeze_frame) {		
field_id	2	uimsbf
reserved	3	bslbf
}		
else if (trick_mode_control == fast_reverse') {		
field_id	2	bslbf
intra_slice_refresh	1	bslbf
frequency_truncation	2	bslbf
else if (trick_mode_control == slow_reverse) {		
rep_cntrl	5	uimsbf
}		
else		
reserved	5	bslbf
}		
if (additional_copy_info_flag == '1') {		
marker_bit	1	bslbf
additional_copy_info	7	bslbf
}		
if (PES_CRC_flag == '1') {		
previous_PES_packet_CRC	16	bslbf
}		
if (PES_extension_flag == '1') {		
PES_private_data_flag	1	bslbf
pack_header_field_flag	1	bslbf
program_packet_sequence_counter_flag	1	bslbf
P-STD_buffer_flag	1	bslbf
reserved	3	bslbf
PES_extension_flag_2	1	bslbf

```
                        if ( PES_private_data_flag == '1' ) {
                                PES_private_data                                128     bslbf
                        }
                        if (pack_header_field_flag == '1' ) {
                                pack_field_length                               8       uimsbf
                                pack_header()
                        }
                        if(program_packet_sequence_counter_flag== '1'){
                                marker_bit                                      1       bslbf
                                program_packet_sequence_counter                 7       uimsbf
                                marker_bit                                      1       bslbf
                                MPEG1_MPEG2_identifier                          1       bslbf
                                original_stuff_length                           6       uimsbf
                        }
                        if ( P-STD_buffer_flag == '1' ) {
                                '01'                                            2       bslbf
                                P-STD_buffer_scale                              1       bslbf
                                P-STD_buffer_size                               13      uimsbf
                        }
                        if ( PES_extension_flag_2 == '1'){
                                marker_bit                                      1       bslbf
                                PES_extension_field_length                      7       uimsbf
                                for(i=0;i<PES_extension_field_length;i++) {
                                        reserved                                8       bslbf
                                }
                        }
                }
                for (i=0;i<N1;i++) {
                        stuffing_byte                                           8       bslbf
                }
                for (i=0;i<N2;i++) {
                        PES_packet_data_byte                                    8       bslbf
                }
        }
        else if ( stream_id == program_stream_map
        || stream_id == private_stream_2
        || stream_id == ECM
        || stream_id == EMM
        || stream_id == program_stream_directory
        || stream_id == DSMCC_stream
        || stream_id == ITU-T Rec. H.222.1 type E stream
        || stream_id == ISO/IEC 14496-1_FlexMux_stream) {
                for ( i=0;i<PES_packet_length;i++) {
                        PES_packet_data_byte                                    8       bslbf
                }
        }
        else if ( stream_id == padding_stream) {
                for ( i=0;i<PES_packet_length;i++) {
                        padding_byte                                            8       bslbf
                }
        }
}
```

“

2) Replace table 2-18 in subclause 2:

“

Table 2-18 -- Stream_id assignments

stream_id	Note	stream coding
1011 1100	1	program_stream_map
1011 1101	2	private_stream_1
1011 1110		padding_stream

1011 1111	3	private_stream_2
110x xxxx		ISO/IEC 13818-3 or ISO/IEC 11172-3 or ISO/IEC 13818-7 or ISO/IEC 14496-3 audio stream number x xxxx
1110 xxxx		ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 or ISO/IEC 14496-2 video stream number xxxx
1111 0000	3	ECM_stream
1111 0001	3	EMM_stream
1111 0010	5	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A or ISO/IEC 13818-6_DSMCC_stream
1111 0011	2	ISO/IEC_13522_stream
1111 0100	6	ITU-T Rec. H.222.1 type A
1111 0101	6	ITU-T Rec. H.222.1 type B
1111 0110	6	ITU-T Rec. H.222.1 type C
1111 0111	6	ITU-T Rec. H.222.1 type D
1111 1000	6	ITU-T Rec. H.222.1 type E
1111 1001	7	ancillary_stream
1111 1010		ISO/IEC14496-1_SL-packetized_stream
1111 1011		ISO/IEC14496-1_FlexMux_stream
1111 1011 … 1111 1110		reserved data stream
1111 1111	4	program_stream_directory

The notation x means that the value '0' or '1' are both permitted and results in the same stream type. The stream number is given by the values taken by the x's.

NOTES

1 PES packets of type program_stream_map have unique syntax specified in 2.5.4.1.

2 PES packets of type private_stream_1 and ISO/IEC_13552_stream follow the same PES packet syntax as those for ITU-T Rec. H.262 | ISO/IEC 13818-2 video and ISO/IEC 13818-3 audio streams.

3 PES packets of type private_stream_2, ECM_stream and EMM_stream are similar to private_stream_1 except no syntax is specified after PES_packet_length field.

4 PES packets of type program_stream_directory have a unique syntax specified in 2.5.5.

5 PES packets of type DSM-CC_stream have a unique syntax specified in ISO/IEC 13818- 6.

6 This stream_id is associated with stream_type 0x09 in Table 2-29.

7 This stream_id is only used in PES packets, which carry data from a Program Stream or an ISO/IEC 11172-1 System Stream, in a Transport Stream (refer to 2.4.3.7).

3) Replace table 2-26 in subclause 2:

"

Table 2-26 -- table_id assignment values

value	description
0x00	program_association_section
0x01	conditional_access_section(CA_section)
0x02	TS_program_map_section
0x03	TS_description_section
0x04	ISO/IEC14496_scene_description_section
0x05	ISO/IEC14496_object_descriptor_section
0x06-0x3F	ITU-T Rec. H.222.0 | ISO/IEC 13818 reserved
0x40-0xFE	User private
0xFF	forbidden

"

4) Replace table 2-29 in subclause 2:

“

Table 2-29 -- Stream type assignments

Value	Description
0x00	ITU-T | ISO/IEC Reserved
0x01	ISO/IEC 11172 Video
0x02	ITU-T Rec. H.262 | ISO/IEC 13818-2 Video or ISO/IEC 11172-2 constrained parameter video stream
0x03	ISO/IEC 11172 Audio
0x04	ISO/IEC 13818-3 Audio
0x05	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 private_sections
0x06	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 PES packets containing private data
0x07	ISO/IEC 13522 MHEG
0x08	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A DSM CC
0x09	ITU-T Rec. H.222.1
0x0A	ISO/IEC 13818-6 type A
0x0B	ISO/IEC 13818-6 type B
0x0C	ISO/IEC 13818-6 type C
0x0D	ISO/IEC 13818-6 type D
0x0E	ISO/IEC 13818-1 auxiliary
0x0F	ISO/IEC 13818-7 Audio with ADTS transport syntax
0x10	ISO/IEC 14496-2 Video
0x11	ISO/IEC 14496-3 Audio
0x12	ISO/IEC 14496 SL-packetized stream or FlexMux stream carried in PES
0x13	ISO/IEC 14496 SL-packetized stream or FlexMux stream carried in ISO/IEC 13818-1 sections
0x14	ISO/IEC 13818-6 Synchronized Download Protocol
0x15-0x7F	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
0x80-0xFF	User Private

“

5) Replace table 2-39 in subclause 2:

“

Table 2-39 -- Program and program element descriptors

descriptor_tag	TS	PS	Identification
0	n/a	n/a	Reserved
1	n/a	n/a	Reserved
2	X	X	video_stream_descriptor
3	X	X	audio_stream_descriptor
4	X	X	hierarchy_descriptor
5	X	X	registration_descriptor
6	X	X	data_stream_alignment_descriptor
7	X	X	target_background_grid_descriptor
8	X	X	video_window_descriptor
9	X	X	CA_descriptor
10	X	X	ISO_639_language_descriptor
11	X	X	system_clock_descriptor
12	X	X	multiplex_buffer_utilization_descriptor
13	X	X	copyright_descriptor
14	X		maximum bitrate descriptor
15	X	X	private data indicator descriptor
16	X	X	smoothing buffer descriptor
17	X		STD_descriptor
18	X	X	IBP descriptor
19-26	X		defined in ISO/IEC 13818-6
27	X	X	MPEG-4_video_descriptor
28	X	X	MPEG-4_audio_descriptor

29	X	X	IOD_descriptor
30	X	X	FMC_descriptor
31	X		SL_descriptor
32	X	X	OCR_ES_ID_descriptor
33	X	X	External_ES_ID_descriptor
34-63	n/a	n/a	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
64-255	n/a	n/a	User Private

“

6) Add subclauses 2.6.36 - 2.6.42:

“

2.6.36 IOD_descriptor

The IOD_descriptor encapsulates the InitialObjectDescriptor structure. The InitialObjecDescriptor allows access to a set of ISO/IEC 14496 streams by identifying the ES_Id values of the ISO/IEC 14496 scene description and ISO/IEC 14496 object descriptor streams. Both the ISO/IEC 14496 scene description stream and ISO/IEC 14496 object descriptor stream contain further information about the ISO/IEC 14496 streams (see P.2.2.4 and P.2.3.5 for a description of the content access procedure). The InitialObjectDescriptor is specified in sub-clause 8.6.3 of ISO/IEC 14496-1.

The IOD_descriptor shall be conveyed in the first descriptor loop immediately following the program_info_length field of the Program Stream Map or TS_program_map_section. More than one IOD_descriptor may be associated to a program.

NOTE - This specification does not specify how the IOD_label may be used by higher level service information to uniquely select one of the MPEG-4 presentations identified by multiple IOD_descriptors.

2.6.36.1 Syntax

Syntax	**No. of bits**	**Mnemonic**
IOD_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
IOD_label	8	**uimsbf**
InitialObjectDescriptor ()		
}		

2.6.36.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x1D to identify the IOD_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

IOD_label -- This 8-bit field shall specify a unique label for the IOD_descriptor.

InitialObjectDescriptor () -- This structure is defined in sub-clause 8.6.3.1 of ISO/IEC 14496-1.

2.6.37 FMC_descriptor

The FMC_descriptor indicates that the ISO/IEC 14496 FlexMux tool has been used to multiplex ISO/IEC 14496-1 SL-packetized streams into a ISO/IEC 14496 FlexMux stream before encapsulating it in a PES stream or a program element consisting of ISO/IEC 14496 sections. The FMC_descriptor associates FlexMux channels to the ES_Id values of the ISO/IEC 14496-1 SL-packetized streams in the ISO/IEC 14496 FlexMux stream.

An FMC_descriptor is required for each elementary_PID or elementary_stream_id, respectively, conveying an ISO/IEC 14496 FlexMux stream. The FMC_descriptor shall be conveyed in the descriptor loop immediately following the ES_info_length field for the corresponding ES within the TS_program_map_section or Program Stream Map.

For each SL_packetized stream, labeled by an ES_ID, that is carried within the FlexMux stream in a single elementary_PID or elementary_stream_id, respectively, one entry in the FMC_descriptor is required in order to identify its FlexMux channel.

2.6.37.1 Syntax

Syntax	No. of bits	Mnemonic
FMC_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
for (i=0; i<descriptor_length; i += 3) {		
ES_ID	**16**	**uimsbf**
FlexMuxChannel	**8**	**uimsbf**
}		
}		

2.6.37.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x1E to identify the FMC_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

ES_ID - This 16-bit field shall specify the identifier of an ISO/IEC 14496-1 SL-packetized stream.

FlexMuxChannel - This 8-bit field shall specify the number of the FlexMux channel used for this SL-packetized stream.

2.6.38 SL_descriptor

The SL_descriptor shall be used when a single ISO/IEC 14496-1 SL-packetized stream is encapsulated in a PES stream. The SL_descriptor associates the ES_ID of this SL-packetized stream to an elementary_PID in case of a Transport Stream or to an elementary_stream_id in case of a Program Stream. This shall be done by conveying a SL_descriptor in the descriptor loop immediately following the ES_info_length field for the corresponding ES within the TS_program_map_section or Program Stream Map.

NOTE - SL_descriptor may be used in a Program Stream. However, only one stream_id exists for ISO/IEC 14496-1 (Systems) elementary streams. In order to associate multiple such MPEG-4 streams to a Program Stream, FlexMux has to be used and signaled appropriately by FMC_descriptor. This limitation does not exist in MPEG-2 Transport Stream as the following descriptor provides unambiguous mapping between an ISO/IEC 14496-1 ES_ID value and an ISO/IEC 13818-1 elementary_PID value.

2.6.38.1 Syntax

Syntax	No. of bits	Mnemonic
SL_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
ES_ID	**16**	**uimsbf**
}		

2.6.38.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x1F to identify the SL_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

ES_ID - This 16-bit field shall specify the identifier of an ISO/IEC 14496-1 SL-packetized stream.

2.6.39 OCR_ES_ID_descriptor

The OCR_ES_ID_descriptor assigns an ES_ID, as defined in ISO/IEC 14496-1, to the entity conveying either SCR or PCR clock reference values. In case of a Program Stream, this entity corresponds to the sequence of SCR values conveyed in the Pack Header. In case of a Transport Stream, this entity corresponds to the elementary stream carrying the Program Clock Reference (PCR), referred by the packet identifier value PCR_PID.

The OCR_ES_ID may be referenced in SLConfigDescriptors, as defined in ISO/IEC 14496-1, in order to signal that the time base of a given ISO/IEC 14496-1 SL-packetized stream is locked to the time base defined by SCR or PCR, as applicable.

The assignment of an ES_ID to the ISO/IEC 13818-1 program element conveying the clock references shall be made by conveying an OCR_ES_ID_descriptor in the first descriptor loop immediately following the program_info_length field of the Program Stream Map or the TS_program_map_section.

2.6.39.1 Syntax

Syntax	No. of bits	Mnemonic
OCR_ES_ID_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
OCR_ES_ID	**16**	**uimsbf**
}		

2.6.39.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x20 to identify the OCR_ES_ID_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

OCR_ES_ID - This 16-bit field shall specify an ES_ID identifier, as defined in ISO/IEC 14496, assigned to the ISO/IEC 13818-1 program element that carries the MPEG-2 time base reference.

2.6.40 External_ES_ID_descriptor

The External_ES_ID_descriptor assigns an ES_ID, as defined in ISO/IEC 14496, to a non-MPEG-4 component of a program. This ES_ID allows reference to the non-MPEG-4 component in the MPEG-4 scene description or, for example, to associate it with an IPMP stream.

The assignment of an ES_ID to a non-MPEG-4 component identified by an elementary_stream_id or elementary_PID shall be made by conveying an External_ES_ID_descriptor in the descriptor loop immediately following the ES_info_length field for the corresponding ES within the Program Stream Map or TS_program_map_section.

2.6.40.1 Syntax

Syntax	No. of bits	Mnemonic
External_ES_ID_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
External_ES_ID	**16**	**uimsbf**
}		

2.6.40.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x21 to identify the External_ES_ID_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

External_ES_ID - This 16-bit field shall specify an ES_ID identifier, as defined in ISO/IEC 14496, assigned to a non-MPEG-4 component of a program.

2.6.41 MPEG-4 Video descriptor

The MPEG-4 Video descriptor provides basic information for identifying the coding parameters of MPEG-4 video elementary streams as described in ISO/IEC 14496-2. Each MPEG-4 video stream carried by MPEG-2 and identified by the MPEG-4 video descriptor shall contain the information required to decode the MPEG-4 video stream; consequently the stream shall contain Visual Object Sequence Headers and Video Object Headers.

The MPEG-4 Video descriptor shall be conveyed in the second descriptor loop immediately following ES_info_length field of the Program Stream Map or TS_program_map_section.

NOTE - Mesh and Face animation cannot be supported without use of scene description (BIFS) and object descriptors as defined in ISO/IEC 14496-1.

2.6.41.1 Syntax

Syntax	No. of bits	Mnemonic
MPEG-4_video_descriptor () {		
descriptor_tag	**8**	**Uimsbf**
descriptor_length	**8**	**Uimsbf**
MPEG-4_visual_profile_and_level	**8**	**Uimsbf**
}		

2.6.41.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x1B to identify the MPEG-4_video_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

MPEG-4_video_profile_and_level - This 8-bit field shall identify the profile and level of the MPEG-4 video stream. This field shall be coded with the same value as the profile_and_level_indication field in the Visual Object Sequence Header specified in ISO/IEC 14496-2.

2.6.42 MPEG-4 Audio descriptor

The MPEG-4 Audio descriptor provides basic information for identifying the coding parameters of MPEG-4 audio elementary streams as described in ISO/IEC 14496-3.

MPEG-4 Audio descriptor shall be conveyed in the second descriptor loop immediately following ES_info_length field of the Program Stream Map or TS_program_map_section.

2.6.42.1 Syntax

Syntax	No. of bits	Mnemonic
MPEG-4_audio_descriptor () {		
descriptor_tag	**8**	**Uimsbf**
descriptor_length	**8**	**Uimsbf**
MPEG-4_audio_profile_and_level	**8**	**Uimsbf**
}		

2.6.42.2 Semantics

descriptor_tag -- This 8-bit field shall be equal to 0x1C to identify the MPEG-4_audio_descriptor.

descriptor_length -- This 8-bit field shall specify the length in bytes of the fields immediately following this field up to the end of the descriptor.

MPEG-4_audio_profile_and_level - This 8-bit field shall identify the profile and level of the MPEG-4 audio stream corresponding to the following table.

Table 2-27 – MPEG-4_audio_profile_and_level assignment values

Value	Description
0x00 - 0x0F	Reserved
0x10	Main profile, level 1
0x11	Main profile, level 2
0x12	Main profile, level 3
0x13	Main profile, level 4
0x14 – 0x17	Reserved
0x18	Scalable Profile, level 1
0x19	Scalable Profile, level 2
0x1A	Scalable Profile, level 3
0x1B	Scalable Profile, level 4
0x1C – 0x1F	Reserved
0x20	Speech profile, level 1
0x21	Speech profile, level 2
0x22 – 0x27	Reserved

0x28	Synthesis profile, level 1
0x29	Synthesis profile, level 2
0x2A	Synthesis profile, level 3
0x2B – 0xFF	Reserved

“

7) Add subclauses 2.4.2.7 and 2.4.2.8:

“

2.4.2.7 STD extensions for carriage of MPEG-4 Video

Size BS(n) of Buffer B(n) : to be defined per profile and level

Rate Rx(n) : 1.2 x Rmax(profile, level)

2.4.2.8 STD extensions for carriage of MPEG-4 Audio

Size BS(n) of Buffer B(n) for AAC as defined in ISO13818-1/AMD6, else BS(n) = 4096 Byte.

Rate Rx(n) for AAC as defined in ISO13818-1/AMD6, else Rx(n) = 2 000 000 Mbit/sec.

“

8) Add Annex P:

“

Annex P

Transport of ISO/IEC 14496 data over ISO/IEC 13818-1 Program Streams or Transport Streams

(This annex does not form an integral part of this Recommendation | International Standard)

P.1 Transport of individual ISO/IEC 14496 elementary streams in ISO/IEC 13818-1 streams

The encapsulation and signaling of individual ISO/IEC 14496-2 or ISO/IEC 14496-3 elementary streams is fully accomplished by the additional descriptors (MPEG-4_video_descriptor and MPEG-4_audio_descriptor) that shall be conveyed in the descriptor loop for the respective entry in either the PMT or PSM in case of Transport Stream or Program Stream, respectively. ISO/IEC 13818-1 does not specify the way how such ISO/IEC 14496 elementary streams are presented in the context of a program.

P.2 Transport of an audiovisual scene represented by ISO/IEC 14496 in ISO/IEC 13818-1 streams

P.2.1 Common specification for Transport Stream and Program Stream

P.2.1.1 Introduction

This annex describes the encapsulation and signaling when an audiovisual scene represented by ISO/IEC 14496 is carried over ISO/IEC 13818-1, either in a Program Stream or in a Transport Stream. The ISO/IEC 14496 content consists of one descriptor, the initialObjectDescriptor and a variable number of streams; the object descriptor stream, scene description stream (carrying either BIFS command or BIFS anim access units), IPMP stream, and audio-visual streams. All of these streams shall be packetized using the sync layer syntax and may optionally be multiplexed using the FlexMux tool, both defined in ISO/IEC 14496-1. SL-packetized streams or FlexMux streams are then encapsulated either in PES or ISO/IEC 14496-1 private sections (see P.2.3.4) prior to TS or PS encapsulation. For some cases a removal of redundancy between PES header and SL header is specified.

P.2.1.2 Constraints on assignment of ES_ID values

Name scoping rules are defined for identifiers in ISO/IEC 14496-1. They may allow the same ES_ID value to be used for two different streams within ISO/IEC 14496 content. When ISO/IEC 14496 content is carried using ISO/IEC 13818-1,such duplicate ES_ID values shall not been permitted.

P.2.1.3 Mapping between SL packets and PES packets

If a single ISO/IEC 14496-1 SL-packetized stream is mapped into a PES stream, one and only one SL packet shall constitute the payload of one PES packet.

PES packets with a payload consisting of a SL packet are identified by stream_id = 0xFA in the PES header. Such PES packets contain the full PES packet header.

A number of fields are redundant between the SL packet header and the PES header. In order to avoid this redundancy the values of these fields in the SL packet header as specified in the SLConfigDescriptor for a given stream (see subclause 10.2.3 in ISO/IEC 14496-1) shall be mapped into PES header fields as specified below and consequently, shall be omitted from the SL packet header. Other fields that cannot be mapped onto PES header field shall be placed at the beginning of the PES packet payload, in their original order. If, according to its related SLConfigDescriptor, an SL packet header contains only elements that are mappable into the PES packet header, then the SL packet header following the PES header may be empty. The original SL packet header can be reconstructed using the parameters in the SLConfigDescriptor and the mapping rules below.

- data_alignment_indicator 1-bit field in the PES header:

This field shall be set to oneto indicate that each PES header is followed immediately by an SL packet header.

- PTS_DTS_flags 2-bit field in the PES header -

PTS_DTS_flags = '00' shall correspond to ISO/IEC 14496-1 compositionTimeStampFlag=0 and ISO/IEC 14496-1 decodingTimeStampFlag=0
PTS_DTS_flags = '10' shall correspond to ISO/IEC 14496-1 compositionTimeStampFlag=1 and ISO/IEC 14496-1 decodingTimeStampFlag=0
PTS_DTS_flags = '11' shall correspond to ISO/IEC 14496-1 compositionTimeStampFlag=1 and ISO/IEC 14496-1 decodingTimeStampFlag=1
PTS_DTS_flags = '00' shall be forbidden

- ES_rate_flag 1 bit flag in the PES header

This field shall convey a copy of the ISO/IEC 14496 instantBitrateFlag

- PTS 33 bit field in the PES header

This field shall corresponds to the ISO/IEC 14496 compositionTimeStamp through the following formulae

$PTS(k) = ((system\_clock\_frequency * t_c(k)) \text{ DIV } 300) \% 2^{33}$.

with

$t_c(k) = (compositionTimeStamp(k)/SL.timeStampResolution + m * 2^{SL.timeStampLength}/SL.timeStampResolution)$

or, equivalently,

$compositionTimeStamp(k) = (SL.timeStampResolution * t_c(k)) \% 2^{SL.timeStampLength}$

with

$t_c(k) = (PTS(k) * 300) / system\_clock\_frequency + n * 2^{33} / system\_clock\_frequency$

$t_c(k)$ indicates the composition time of composition unit cu(k) .

$compositionTimeStamp(k)$ is the time stamp associated to $t_c(k)$ according to the settings in the SLConfigDescriptor for a given SL-packetized stream.

PTS(k) is the ISO/IEC 13818-1 presentation time stamp associated to $t_c(k)$ expressed on the basis of system_clock_frequency and with 33 bit length as defined in this specification.

- DTS 33-bit field in the PES header

This field shall corresponds to the ISO/IEC 14496 decodingTimeStamp through the following formula

$DTS(k) = ((system\_clock\_frequency * t_d(k)) \text{ DIV } 300) \% 2^{33}$.

with

$t_d(k) = (decodingTimeStamp(k)/SL.timeStampResolution + m * 2^{SL.timeStampLength}/SL.timeStampResolution)$

or, equivalently,

`decodingTimeStamp(k)` = (`SL.timeStampResolution` * $t_d(k)$) % $2^{SL.timeStampLength}$

with

$t_d(k)$ = (DTS(k) * 300) / system_clock_frequency + n * 2^{33} / system_clock_frequency

$t_c(k)$ indicates the decoding time of access unit au(k) .

`decodingTimeStamp(k)` is the time stamp associated to $t_d(k)$ according to the settings in the SLConfigDescriptor for a given SL-packetized stream.

DTS(k) is the ISO/IEC 13818-1 decoding time stamp associated to $t_d(k)$ expressed on the basis of system_clock_frequency and with 33 bit length as defined in this specification.

- ES_rate 22-bit field in the PES header
- This bit shall represent the ISO/IEC 14996 instantBitrate field value (in bits per second) divided by / 400

P.2.1.4 Mapping between FlexMux packets and PES packets

The ISO/IEC 14496 FlexMux tool may be used to map multiple ISO/IEC 14496-1 SL-packetized streams into a PES stream. The payload of FlexMux packets shall consist of proper SL packets as specified in ISO/IEC 14496-1. An integer number of FlexMux packets shall constitute the payload of one PES packet, i.e., the payload of a PES packet shall always start with a FlexMux packet header and shall end with the last byte of a FlexMux packet.

PES packets with a payload consisting of FlexMux packets are identified by stream_id = 0xFB in the PES header. Such PES packets shall not include the PES packet header.

P.2.2 Transport of ISO/IEC 14496 data in an ISO/IEC 13818-1 Program Stream

P.2.2.1 Overview

A Program Stream may only contain one program. ISO/IEC 14496 data can be conveyed in addition to the already defined stream types for such a program as specified below. As a special case, it is also possible that a Program Stream carry only ISO/IEC 14496 data.

P.2.2.2 Initial Object Descriptor

The ISO/IEC 14496 Initial Object Descriptor serves as the initial access point to all other ISO/IEC 14496program elements. It shall be conveyed in the IOD_descriptor that is located in the first descriptor loop of the Program Stream Map. It contains ES_Descriptors identifying the scene description and object descriptor streams that form part of this program. It may also contain ES_Descriptors identifying one or more associated IPMP or OCI streams. Identification of streams is done by means of ES_IDs as specified in clause 8 of ISO/IEC 14496-1.

P.2.2.3 Encapsulation of ISO/IEC 14496 streams

All ISO/IEC 14496 elementary stream types shall be encapsulated as SL-packetized streams. The stream_id (0xFA) field value shall be used to signal ISO/IEC 14496 SL-packetized streams encapsulated in PES (see P.2.1.3). ISO/IEC 14496 content consisting of multiple streams shall be multiplexed into PES packets identified by the stream_id 0xFB using FlexMux. The ES_ID and the associated FlexMux channel for each elementary stream within the multiplex shall be signaled in an FMC_descriptor located in the second descriptor loop of the Program Stream Map entry for this stream_id. The object descriptor and scene description streams identified as per P.2.2.2 form part of this multiplexed stream and can be located by looking up their FlexMux channels in the FMC_descriptor.

P.2.2.4 Content access procedure for ISO/IEC 14496 program components within a Program Stream

The following provides a reference receiver acquisition procedure for accessing ISO/IEC 14496 program elements in an ISO/IEC 13818-1 Program Stream:

1. Acquire the Program Stream Map
2. Identify the IOD_Descriptor in the first descriptor loop
3. Identify the ES_IDs of the object descriptor, scene description and other streams described within the InitialObjectDescriptor
4. Acquire the SL_descriptor and FMC_descriptor in the second descriptor loop for elementary_stream_ids 0xFA and

0xFB, as applicable.

5. Generate from these descriptors a stream map table between ES_IDs and associated elementary_stream_id plus FlexMux channel, if applicable.
6. Locate the object descriptor stream using its ES_ID and the stream map table.
7. Locate other streams described in the InitialObjectDescriptor using their ES_ID and the stream map table.
8. Continuously monitor the object descriptor stream and identify ES_IDs of additional streams.
9. Locate the additional streams using their ES_ID and the stream map table.

NOTE - The Program Stream Map shall be updated as specified in 2.5.4.2 and in accordance with updates made to the ISO/IEC 14496 SL_descriptor or FMC_descriptor descriptors, respectively.

Figure P-1 gives an example of ISO/IEC 14496 content in a Program Stream, consisting of object descriptor stream, scene description stream (BIFS-Command), BIFS-Anim stream and IPMP stream. All the streams are multiplexed in a single FlexMux stream.

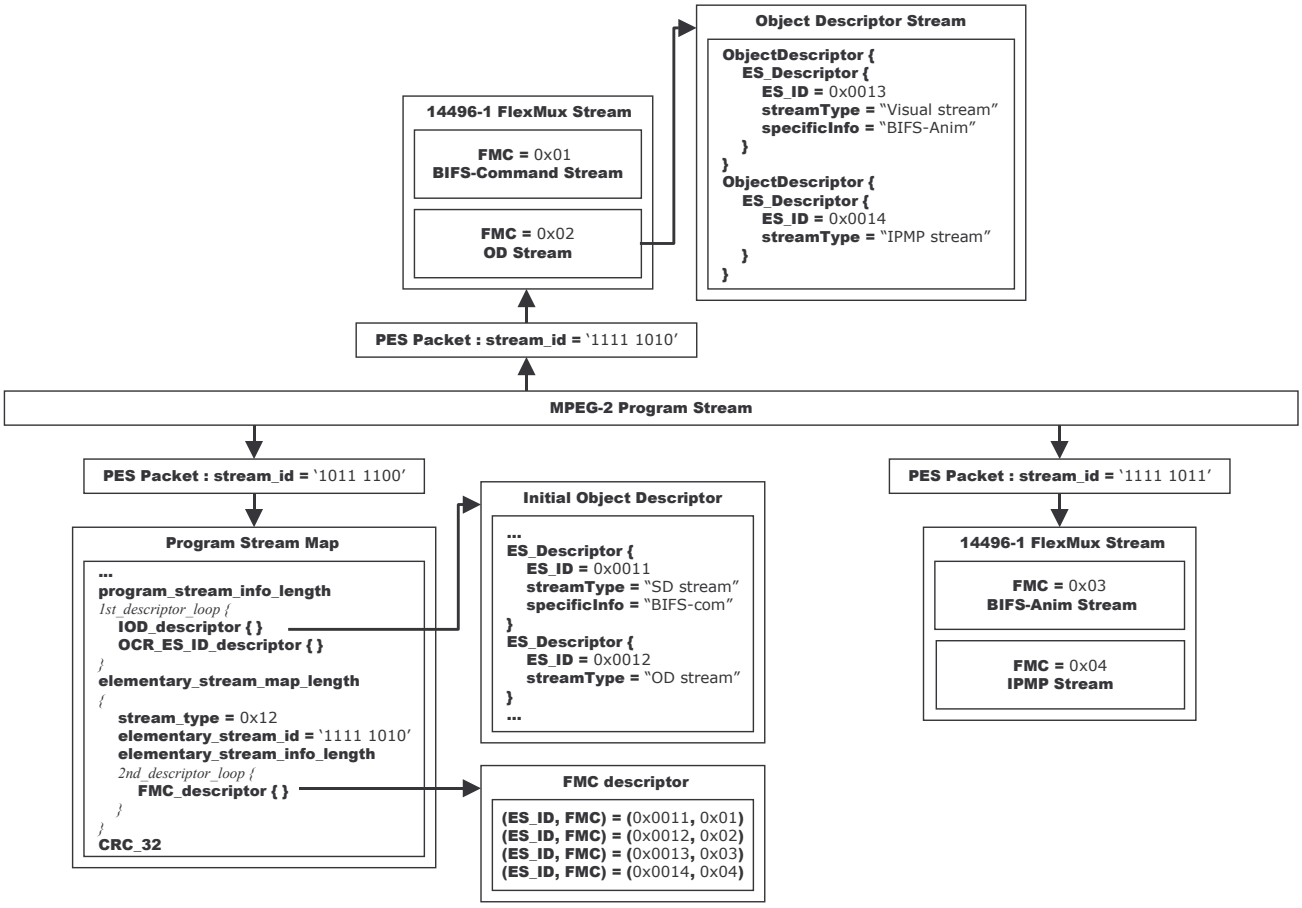

Figure P-1 - Example of ISO/IEC 14496 content in a Program Stream

P.2.3 Transport of ISO/IEC 14496 content in an ISO/IEC 13818-1 Transport Stream

P.2.3.1 Overview

A Transport Stream may contain one or more programs, each described by a Program Map Table. ISO/IEC 14496 content can be conveyed in addition to the already defined stream types for such a program as specified below. Elements of the ISO/IEC 14496 content may be conveyed in one or more PIDs within a transport stream. As a special case, it is possible that a program within a Transport Stream consists only of ISO/IEC 14496 program elements.

P.2.3.2 Initial Object Descriptor

The ISO/IEC 14496 Initial Object Descriptor serves as the initial access point to all other ISO/IEC 14496program elements. It shall be conveyed in the IOD_descriptor located in the first descriptor loop of the TS_program_map_section for the program to which the ISO/IEC 14496 content is associated. It contains ES_Descriptors identifying the scene description and object descriptor streams that form part of this program. It may also contain ES_Descriptors identifying one or more associated IPMP or OCI streams. Identification of streams is done by means of ES_IDs as specified in clause 8 of ISO/IEC 14496-1.

P.2.3.3 Encapsulation of ISO/IEC 14496 streams

All types of ISO/IEC 14496 elementary streams shall be encapsulated as SL-packetized streams. Object Descriptor streams and BIFS-Command SL-packetized streams shall be encapsulated either in PES (see P.2.1.3) or in ISO/IEC 13818-1 private sections (see P.2.3.4). All other stream types, including BIFS-Anim, IPMP and audio-visual elementary streams, shall always be encapsulated in PES. In case of PES encapsulation, under some conditions, the SL packet header may be collapsed into fields of the PES header structure, resulting in a null SL header at the beginning of the PES packet payload.

PES packets carrying ISO/IEC 14496 SL packetized streams shall be identified by the stream_id field value 0xFA. PES packets carrying multiplexed ISO/IEC 14496 SL-packetized streams, using FlexMux, shall be identified by the stream_id field value 0xFB. The one or more elementary_PIDs that convey a (set of) elementary stream using PES syntax shall be identified with the stream_type field value 0x12 in the TS_program_map_section. The ES_ID and the associated FlexMux channel (if applicable) shall be signaled by an SL_descriptor or an FMC_descriptor, respectively, in the second descriptor loop of the TS_program_map_section for this PID.

The elementary streams referenced each by a distinct elementary_PID field value and conveying an Object Descriptor stream or a BIFS-Command stream using ISO/IEC 13818-1 private sections shall be identified with the stream_type field value 0x13 in the TS_program_map_section. FlexMux may be used if multiple SL packets of a stream are multiplexed into a single section. The ES_ID and the FlexMux channel (if applicable) of the encapsulated streams shall be signaled by an SL_descriptor or an FMC_descriptor, respectively, in the second descriptor loop of the TS_program_map_section for this elementary_PID value.

P.2.3.4 Section syntax for transport of ISO/IEC14496 streams

Table P-1 shows the syntax used for sections conveying ISO/IEC 14496 program elements, qualified by the table_id as either scene description or object descriptor stream data.

Table P-1 -- Section syntax for transport of ISO/IEC14496 streams

Syntax	No. of bits	Mnemonic
ISO/IEC14496_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
private_indicator	**1**	**bslbf**
reserved	**2**	**bslbf**
private_section_length	**12**	**uimsbf**
table_id_extension	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
if (PMT_hasSLdescriptor(current_PID)) {		
SL_packet()		
} else {		
for (i=1; i<N; i++)		
FlexMuxPacket()		
}		
CRC_32	**32**	**rpchof**
}		

table_id – This 8-bit field shall be set to '0x05' if it carries an ISO/IEC 14496 object descriptor stream or to '0x04' if it carries a ISO/IEC 14496 BIFS command stream.

section_syntax _indicator – This 1-bit field shall be set to '1'.

private_indicator - This is 1-bit field shall not be specified by this specification

private_section_length - This 12-bit field shall specify the number of remaining bytes in the section immediately following the private_section_length field up to the end of the ISO/IEC14496_section. The value of this field shall not exceed 4093 (0xFFD)

table_id_extension - This 16-bit field shall not be specified by this specification.

version_number - This 5-bit field shall represent the version number of the ISO/IEC14496_section. The version number

shall be incremented by 1 modulo 32 when a change in the information carried with the section occurs.

current_next_indicator - This 1-bit field shall be set to 1.

section_number - This 8-bit field shall represent the number of the ISO/IEC14496_section section. The section_number of the first section in a ISO/IEC14496_section section shall be 0x00. The section_number shall b incremented by 1 with each additional section in this ISO/IEC14496 table.

last_section_number - This 8-bit field shall specify the number of the last section of the ISO/IEC 14496 table of which this section is a part.

PMT_hasSLdescriptor(current_PID) – a pseudo function that shall return '1' if the PMT entry for the PID that conveys this ISO/IEC14496_section contains an SL_descriptor.

SL_packet() – a sync layer packet as specified in subclause 10.2.2 of ISO/IEC 14496-1.

FlexMuxPacket() – a FlexMux packet as specified in subclause 11.2.4 of ISO/IEC 14496-1.

CRC_32 - This 32-bit field shall contain the CRC value that gives a zero output of the registers in the decoder defined in Annex A of ISO/IEC 13818-1 after processing the entire ISO/IEC14496_section section.

The payload data bytes of an ISO/IEC14496_section section shall be defined as the eight bytes at the beginning of the section (from the field table_id included to the last_section_number field included) and the four CRC32 bytes at the end of the section.

P.2.3.5 Content access procedure for ISO/IEC 14496 program components within a Transport Stream

The following provides a reference receiver acquisition procedure for accessing ISO/IEC 14496 program elements in an ISO/IEC 13818-1 Transport Stream:

1. Acquire the Program Map Table for the desired program
2. Identify the IOD_Descriptor in the first descriptor loop
3. Identify the ES_IDs of the object descriptor, scene description and other streams described within the InitialObjectDescriptor
4. Acquire the set of all SL_descriptors and FMC_descriptors present in the second descriptor loop for any of the elementary_PIDs.
5. Generate from these descriptors a stream map table between ES_IDs and associated elementary_PID plus FlexMux channel, if applicable.
6. Locate the object descriptor stream using its ES_ID and the stream map table.
7. Locate other streams described in the InitialObjectDescriptor using their ES_ID and the stream map table.
8. Continuously monitor the object descriptor stream and identify ES_IDs of additional streams.
9. Locate the additional streams using their ES_ID and the stream map table.

NOTE - The Program Map Table shall be updated as specified in 2.4.4.9 and in accordance with the updates made to any of the ISO/IEC 14496 SL_descriptors or ISO/IEC 14496 FMC_descriptors.

Figure P-2 gives an example of ISO/IEC 14496 program elements in a Transport Stream, consisting of object descriptor stream, scene description stream (BIFS-Command), BIFS-Anim and IPMP elementary streams. BIFS-Command and OD stream are conveyed by means of private sections, while BIFS-Anim and IPMP elementary streams are conveyed in PES packets referenced by two distinct elementary_PIDvalues, without the use of the ISO/IEC 14496 FlexMux tool.

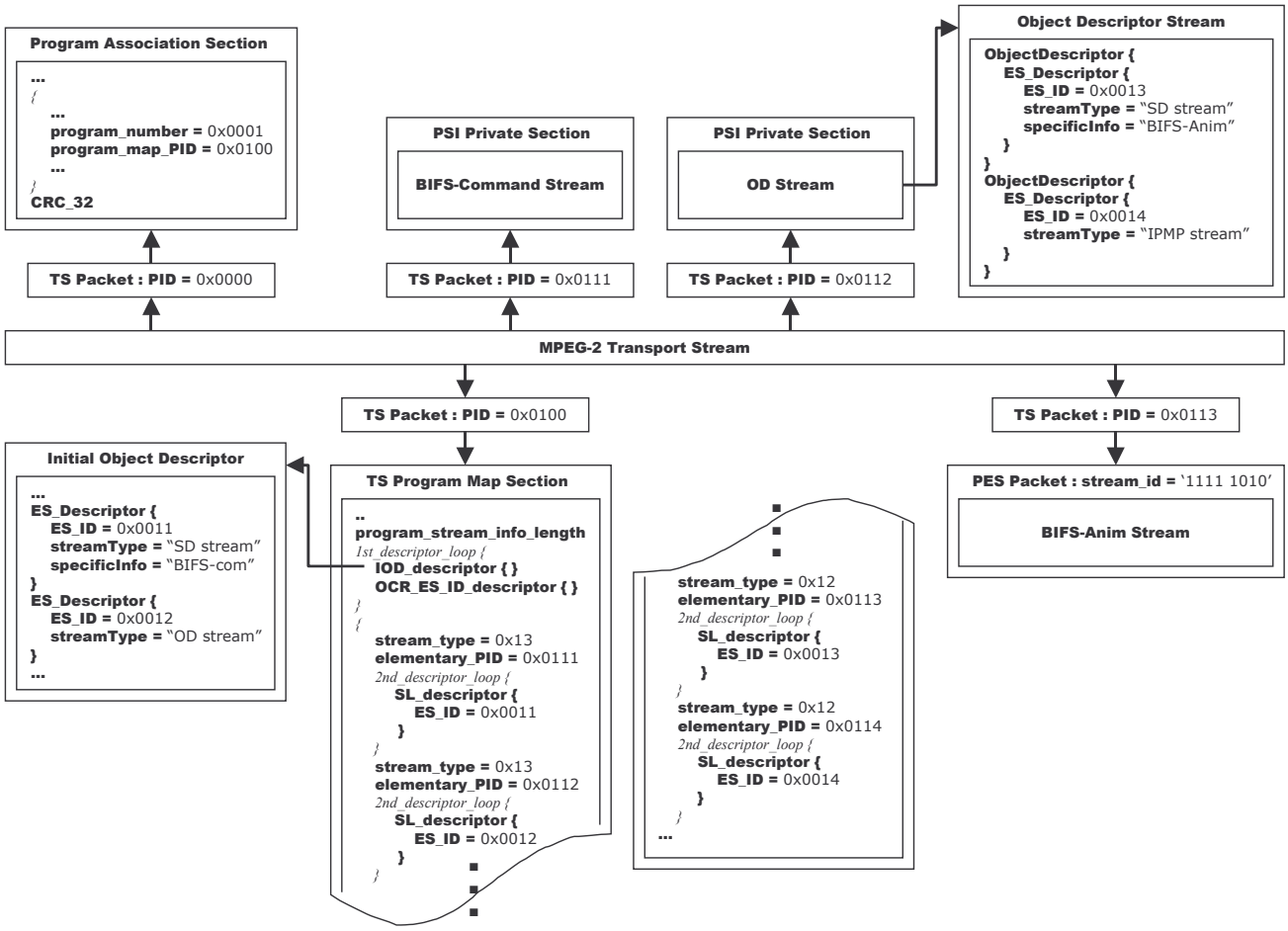

Figure P-2 - Example of ISO/IEC 14496 content in a Transport Stream

P.3 Extensions of the System Target Decoder model

P.3.1 T-STD model for ISO/IEC 14496 program elements

P.3.1.1 Introduction

Figure P-3 shows the Transport System Target Decoder for delivery of ISO/IEC 14496 program elements encapsulated in ISO/IEC 13818-1 Transport Streams.

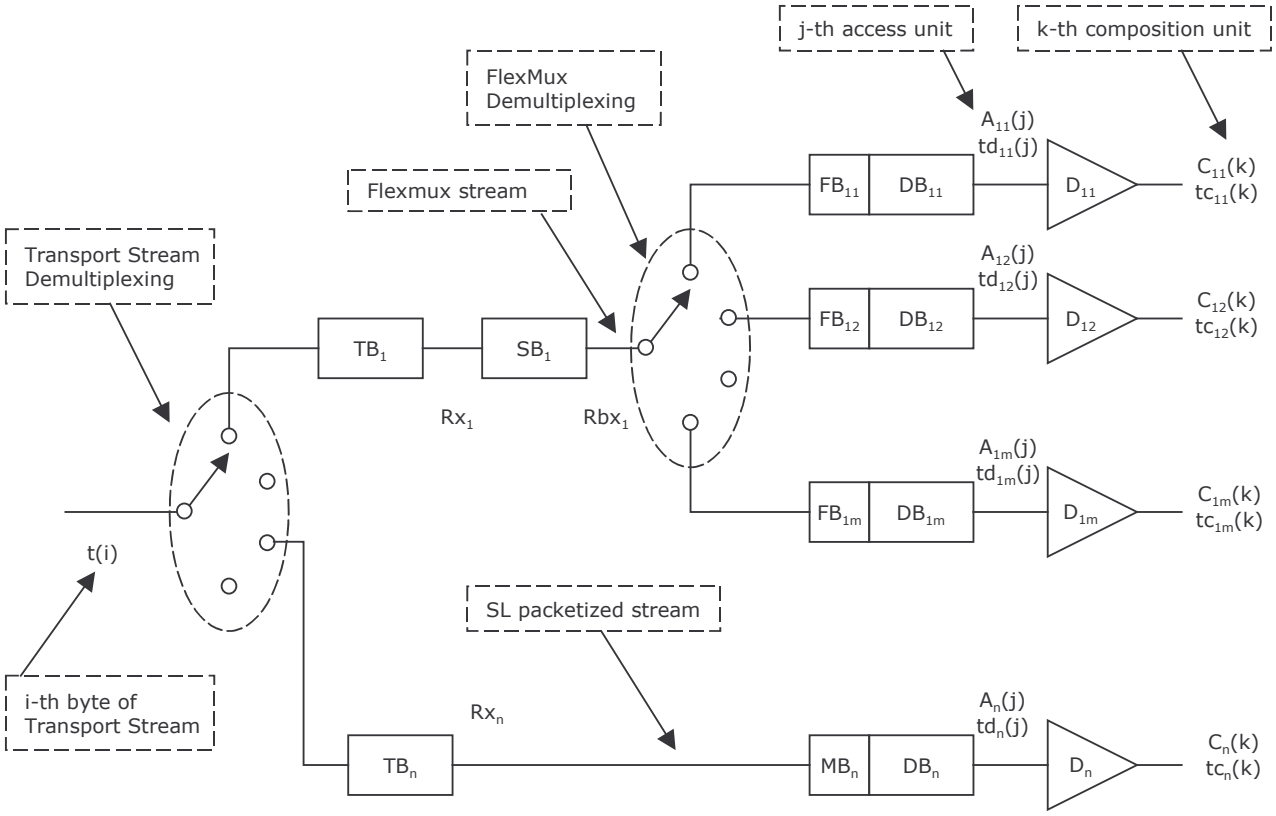

Figure P-3 - T-STD model for ISO/IEC 14496 content

The following notation is used in Figure P-3:

TB_n is the transport buffer.
SB_n is the smoothing buffer for FlexMux stream n.
FB_{nk} is the FlexMux buffer for elementary stream n and FlexMux channel k.
MB_n is the multiplex buffer for elementary stream n (only applicable if FlexMux is not used).
DB_{nk} Is the decoder buffer for elementary stream n and FlexMux channel k
DB_n is the decoder buffer for elementary stream n.
D_n is the decoder for elementary stream n.
Rx_n is the rate at which data are removed from TB_n.
Rfx_n is the rate at which data are removed from FB_n.
$A_n(j)$ is the j^{th} access unit in elementary stream n. An(j) is indexed in decoding order.
$Td_n(j)$ is the decoding time, measured in seconds, in the system target decoder of the jth access unit in elementary stream n.
$C_n(k)$ is the k^{th} composition unit in elementary stream n. $C_n(k)$ results from decoding $A_n(j)$. $C_n(k)$ is indexed in composition order.
$tc_n(k)$ is the composition time, measured in seconds, in the system target decoder of the k^{th} composition unit in elementary stream n.
t(i) indicates the time in seconds at which the i^{th} byte of the Transport Stream enters the system target decoder.

P.3.1.2 Definition of the FlexMux buffer, FB_{nk}

The FlexMux buffer is defined in order to absorb the difference of bitrates between the output of smoothing buffer SB_n and the input of decoder buffers DB_{nk}.. All data that enters SB_n leaves it. If there is PES packet payload data or ISO/IEC 14496 section payload data in SB_n and buffer FB_{nk} is not full, the data is transferred from SB_n to FB_{nk} at a rate equal to Rbx_n . If buffer FB_{nk} is full, data is not removed from SB_n . When a byte of payload data is transferred from SB_n to FB_{nk} , all PES packet header bytes (6 bytes corresponding to packet_start_code_prefix, stream_id and PES_packet_length fields) and FlexMux packet header bytes or ISO/IEC 14496 section header bytes (12 bytes) and FlexMux packet header bytes that are in SB_n and immediately precede or follow that payload byte are instantaneously removed and discarded. Discarded bytes may be used to control the system or to verify the integrity of the data. When there is no PES packet payload data or ISO/IEC 14496 section payload data present in SB_n , no data is removed from SB_n . All PES packet payload data bytes enter FB_{nk} instantaneously upon leaving SB_n . The buffer SB_n shall not be allowed to overflow.

The size of the FlexMux buffer FB_n , FBS_n shall be fixed to *TBD*.

P.3.1.3 Definition of the decoder buffer, DB_{nk}

When a byte is transferred from FB_{nk} to DB_{nk}, all SL packet header bytes that are in FB_n and immediately precede that byte are instantaneously removed and discarded. All that data that enters DB_{nk} leaves it. All SL packet payload data bytes enter DB_n instantaneously upon leaving FB_{nk}. For each $DB_{nk,}$ all data for the SL PDU that has been in the buffer longest and any stuffing bytes that immediately precede it that are present in the buffer at the time td_{nk} are removed instantaneously at time $td_{nk.}$ The decoding time time td_{nk} is specified in the DTS or PTS fields. The combination of the buffers FB_n and DB_n shall not be allowed to overflow.

The decoder buffer DB_n stores SL packets (both header and payload).

The decoder buffer size DBS_n is defined as follows :

$$DBS_n = BS_{oh} + BS_{dec}$$

where the size of the SL packet overhead buffer, BS_{oh}, is *TBD*,

and the size of the access unit decoding buffer, BS_{dec}, is given by the bufferSizeDB of the decoderConfigDescriptor defined in ISO/IEC 14496-1.

The total amount of the bufferSizeDB is limited as follows:

For all visual streams in a program

$$bufferSizeDB_{v1} + bufferSizeDB_{v2} + ... + bufferSizeDB_{vn} <= VBV_{max}[profile,level]$$

VBV_{max}[profile,level] is specified according to the profile and level which can be found in Table N-1 of ISO/IEC 14496-2. For the profiles and levels that define no VBV_{max} value, the total amount of the $bufferSizeDB_n$ is user defined.

For all audio streams in a program

$$bufferSizeDB_{a1} + bufferSizeDB_{a2} + ... + bufferSizeDB_{an} <= TBD \text{ bytes}$$

For BIFS command and object descriptor streams in a program

$$bufferSizeDB_{s1} + bufferSizeDB_{s2} + ... + bufferSizeDB_{sn} <= TBD \text{ bytes}$$

For other systems stream (e.g., animation stream and IPMP stream) in a program

$$bufferSizeDB_{o1} + bufferSizeDB_{o2} + ... + bufferSizeDB_{on} <= TBD \text{ bytes}$$

P.3.1.4 Definition of the rates Rx_n and Rbx_n

All bytes that enter the transport buffer TB_n are removed at the rate Rx_n unless there is no data in TB_n.

All bytes that enter the smoothing buffer SB_n are removed at the rate Rbx_n unless there is no data in SB_n.

There are two possible methods to determine the leak rates Rbx_n and Rx_n.

fixed leak rate

The rate Rx_n for the flexmux stream is defined as follows :

$$Rx_n = Rfxv + Rfxa + Rfxs$$

where the video rate Rfxv is equal to 1.2 x R_{max}[profile,level] if the flexmux stream contains one or more visual streams. Otherwise Rfxv is equal to zero. R_{max}[profile,level] is specified according to the profile and level which can be found in Table N-1 of ISO/IEC 14496-2. For the profiles and levels that define no R_{max} value, Rfxv is user defined.

and the audio rate Rfxa is equal to *TBD* bits per second if the flexmux stream contains one or more audio streams. Otherwise Rfxa is equal to zero.

and the system rate Rfxs is equal to *TBD* bits per second if the flexmux stream contains one or more system streams. Otherwise Rfxs is equal to zero.

The rate Rx_n for the SL packetized stream and the rate Rfx_n are defined as follows :

For visual stream

Rx_n or Rfx_n = 1.2 x R_{max}[profile,level]

where, R_{max}[profile,level] is specified according to the profile and level which can be found in Table N-1 of ISO/IEC 14496-2. For the profiles and levels that define no R_{max} value, Rx_n and Rfx_n is user defined.

For audio stream

Rx_n or Rfx_n = *TBD* bits per second

For BIFS command and object descriptor stream

Rx_n = *TBD* bits per second

For other systems data streams (e.g., animation stream and IPMP stream)

Rx_n or Rfx_n = *TBD* bits per second

maxBitrate of decoderConfigDescriptor

The rate Rx_n for the flexmux stream is defined as follows :

Rx_n = Rfxv + Rfxa + Rfxs

where the visual rate Rfxv is given by :

Rfxv = Min { 1.2 x R_{max}[profile,level] , Rfx_{v1} + Rfx_{v2} + ... + Rfx_{vn} }

if the flexmux stream does not have any visual streams, Rfxv is equal to zero. R_{max}[profile,level] is specified according to the profile and level which can be found in Table N-1 of ISO/IEC 14496-2. For the profiles and levels that define no R_{max} value, the right condition in the above equation shall be used.

and the audio rate Rfxa is given by :

Rfxa = Rfx_{a1} + Rfx_{a2} + ... + Rfx_{an}

and the system rate Rfxs is given by :

Rfxs = Rfx_{s1} + Rfx_{s2} + ... + Rfx_{sn}

The rate Rx_n for the SL packetized stream and the rate Rfx_n are defined as follows :

Rx_n or Rfx_n = 1.2 x maxBitrate

The maxBitrate is the field of the decoderConfigDescriptor defined in ISO/IEC 14496-1. The following limitation applies for the maxBitrate.

For visual streams

maxBitrate <= R_{max}[profile,level]

where, R_{max}[profile,level] is specified according to the profile and level which can be found in Table N-1 of ISO/IEC 14496-2. For the profiles and levels that define no R_{max} value, upper bound of the maxBitrate is user defined.

For audio stream

maxBitrate <= *TBD* bits per second

For BIFS command and object descriptor stream

maxBitrate <= *TBD* bits per second

For other systems data streams (e.g., animation stream and IPMP stream)

maxBitrate <= *TBD* bits per second

The structure of the STD model applicable for the hierarchical multiplexing is depicted in Figure P-3.

P.3.2 P-STD model for ISO/IEC 14496 content

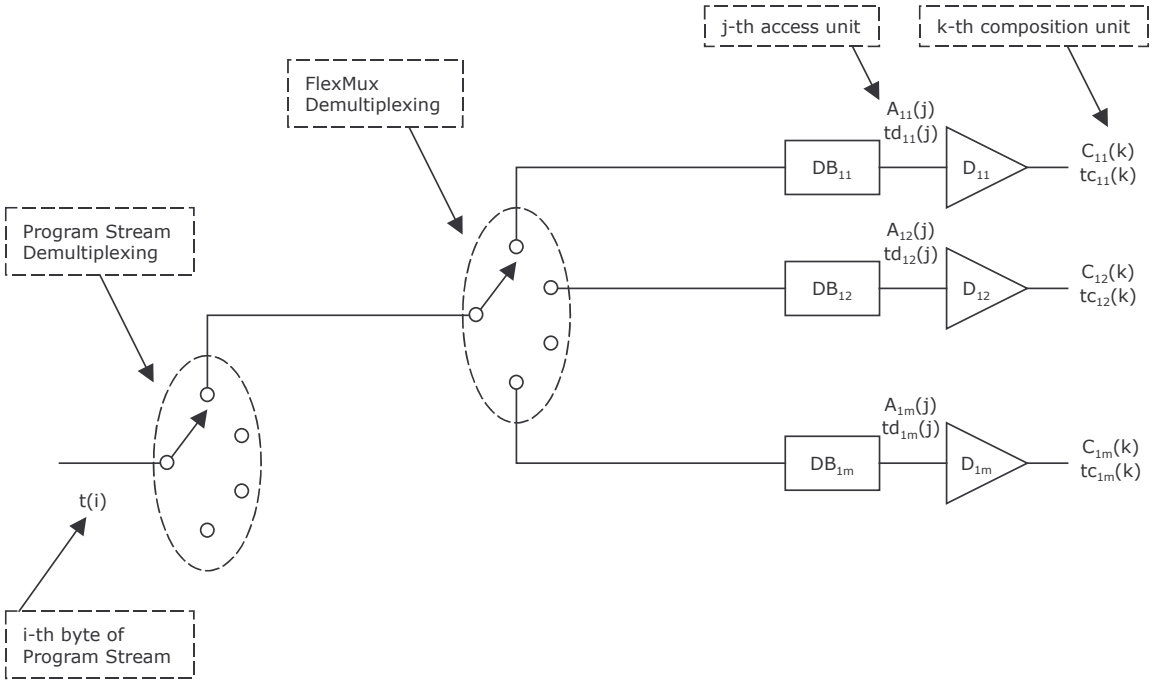

Figure P-4 - P-STD model for MPEG-4 Systems stream

Figure 2 shows the STD model when MPEG-4 systems data are transmitted in MPEG-2 Program Stream. Transfer of byte i from the STD input to DBn is instantaneous.

Bytes present in the pack header, PES packet header and flexmux packet header are not delivered to the decoder buffer. The decoder buffer DB_n is defined in P.3.1.3.
“